NASA Contractor Report 189685
ICASE Report No. 92–35

ICASE

UNSTRUCTURED MESH ALGORITHMS FOR AERODYNAMIC CALCULATIONS

D. J. Mavriplis

Contract Nos. NAS1–18605 and NAS1–19480
July 1992

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665–5225

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

UNSTRUCTURED MESH ALGORITHMS FOR AERODYNAMIC CALCULATIONS

D. J. Mavriplis[1]
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23665-5225

ABSTRACT

The use of unstructured mesh techniques for solving complex aerodynamic flows is discussed. The principle advantages of unstructured mesh strategies, as they relate to complex geometries, adaptive meshing capabilities, and parallel processing are emphasized. The various aspects required for the efficient and accurate solution of aerodynamic flows are addressed. These include mesh generation, mesh adaptivity, solution algorithms, convergence acceleration and turbulence modeling. Computations of viscous turbulent two-dimensional flows and inviscid three-dimensional flows about complex configurations are demonstrated. Remaining obstacles and directions for future research are also outlined.

[1]This research was supported by the National Aeronautics and Space Administration under NASA Contract Nos. NAS1-18605 and NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665.

1. INTRODUCTION

Over the last decade, much attention has been devoted to the development and use of unstructured mesh methodologies within the research community. This enthusiasm however, has not always been shared by the applications and industrial community. The promise of easily enabling the discretization of complex geometries has been counterbalanced by questions of accuracy and efficiency. Furthermore, the dearth of results concerning viscous flow calculations using unstructured meshes has produced skepticism concerning the value of unstructured mesh techniques for practical aerodynamic calculations.

There is no doubt that block-structured techniques have proved extremely effective in discretizing very complex geometries. However, unstructured grid techniques offer additional inherent advantages which may not at first appear evident. The possibility of easily performing adaptive meshing is perhaps the largest advantage of unstructured grid methods. In fact, the implementation of adaptive meshing techniques for structured meshes has generally been found to incur unstructured-mesh type overheads [1]. Furthermore, although unstructured grid data-sets are irregular, they are homogeneous (as opposed to block structured grids where differentiation between block boundaries and interiors must be made). One of the consequences of this property is that unstructured-mesh type solvers are relatively easily parallelizable. While unstructured mesh solvers always incur additional memory and CPU-time overheads due to the random nature of their data-sets, large gains in efficiency can be obtained by careful choices of data-structures, and by resorting to more efficient implicit or multi-level solution procedures. When combined with adaptive meshing and parallelization, these can result in truly competitive solution procedures.

In the following sections, a brief outline of some of the various approaches currently in use in unstructured mesh solution strategies is given, and the various advantages and trade-offs of each method are discussed. This is followed by a set of illustrative example solutions taken from the author's own work, which include two-dimensional viscous flows and three-dimensional inviscid flow solutions on sequential and parallel machine architectures.

2. DISCRETIZATIONS

2.1. Vertex Based and Cell-Centered Schemes

The first choice which arises in the context of unstructured mesh discretizations is the issue of cell-centered versus vertex-based schemes. Unlike the situation for structured grids, where the differences between these two types of schemes consist principally of different boundary condition treatments, the situation for unstructured meshes is quite different. Whereas a hexahedral structured mesh contains the same number of cells as vertices (asymptotically neglecting boundary effects), an unstructured tetrahedral mesh with N vertices contains α N tetrahedral cells, where α is usually between 5 or 6 (there are twice as many triangles as vertices in two dimensions). Thus a cell centered scheme for unstructured meshes requires the solution of 5 to 6 times more unknowns than a vertex based scheme operating on the same grid. Therefore, a cell-centered scheme can be expected to incur substantially higher memory and CPU overheads on a given grid than a vertex scheme.

On the other hand, the solution of a larger number of unknowns would suggest that higher accuracy may be achieved on the same grid using a cell-centered scheme. If one visualizes an unstructured mesh as a simple graph (i.e. a collection of vertices joined together by a set of edges or links), then the vertex scheme is seen to operate on the original graph of the

grid, and the cell-centered scheme on a dual graph, i.e. the dual obtained by placing a vertex at the center of each tetrahedron, and associating a link with each triangular face of the tetrahedra, thus joining neighboring cell centers. The original graph thus contains N vertices and $(\alpha + 1)N$ links, whereas the dual graph contains αN vertices and $2\alpha N$ links. In the original graph, the degree of each vertex (number of incident links) is variable, but averages out to $2(\alpha + 1)$. In the dual graph, the degree of each vertex is fixed and equal to 4. By comparison, the degree of each vertex in a hexahedral mesh is 6, for both cell centered and vertex schemes.

Thus, although the vertex scheme contains 5 to 6 times less unknowns on a given grid than the cell-centered scheme, these vertices are more tightly coupled than those of the cell-centered scheme. This in turn suggests that, although there are less vertices, the discretization at each vertex may be more accurate than in the cell-centered scheme.

Practical evidence indicates that on a given grid, for inviscid flows, cell-centered schemes appear to yield somewhat higher accuracy than vertex schemes. The crucial question is thus whether this perceived increase in accuracy is sufficient to overcome the substantial memory and CPU overheads incurred by the cell centered schemes. Unfortunately, few direct comparisons have been made between vertex and cell-centered unstructured schemes, and these have usually been hindered by the use of different discretization schemes and/or different grids. This is an area which should be further investigated in the future. Furthermore, the above discussion illustrates the dangers of comparing vertex and cell-centered unstructured schemes with each other or with structured grid solvers based on the number of unknowns, without regard for the amount of connectivity between the unknowns.

2.2. Central Difference and Upwind Schemes

The same benefits, trade-offs and controversies exist concerning the use of central-difference schemes (with additional artificial dissipation) and upwind schemes on unstructured meshes as in the structured mesh context. The equivalent of a central-difference discretization can be formulated on an unstructured mesh as a Galerkin finite-element discretization where the variables are stored at the vertices of the mesh and the fluxes are assumed to vary piecewise linearly over the cells of the mesh [2,3]. This results in a nearest neighbor stencil which is non-dissipative. Additional dissipative terms are thus constructed as a blend of a Laplacian and a biharmonic operator, which correspond to the second and fourth differences employed in the structured mesh context for damping out oscillations in the vicinity of shocks, and in smooth regions of the flow, respectively. These schemes are simple to construct and relatively inexpensive to compute. Furthermore, they are easily linearizable for use with implicit schemes [4]. The explicit control over the amount of dissipation in the scheme can be viewed either as an advantage (additional control) or as a disadvantage (additional input parameters). Additional improvements to central difference schemes are possible, such as the use of matrix valued dissipation [5], which attempts to scale the dissipative terms among the various wave components of the governing equations.

Upwind schemes are more complex and expensive than simple Galerkin finite-element schemes, but offer the possibility of capturing shocks with higher resolution. The amount of dissipation is automatically determined by the scheme and split appropriately between the various wave components of the governing equations. The most successful upwind schemes for unstructured meshes have been those based on flux differencing [6,7,8]. The introduction of multi-dimensional reconstruction for the extension to second-order schemes [8] has put these methods on a more solid theoretical foundation. On the other hand, one-dimensional Rieman

solvers are still required, although much research is currently devoted to developing truly multi-dimensional upwind schemes [9]. The use of limiters with such schemes, which is required in the presence of shock waves, has often been found to inhibit convergence to steady-state.

Higher order schemes have also been investigated by a number of researchers (see for example [10,11]). Such schemes offer the possibility of resolving complex flows in a more efficient manner using a more accurate (and expensive) representation of the data on a smaller number of mesh points. Although few if any such schemes are routinely used today, they will probably play an important role in the future for the solution of high Reynolds number viscous flows which presently require the use of tens of millions of grid points. It is interesting to note that in the structures field, unstructured higher order discretizations are the method of choice.

In the context of unstructured meshes, the increased cost of upwind or higher order methods must be weighed against the cost of retaining an inexpensive discretization and making use of adaptive meshing techniques, which constitute one of the main advantages of unstructured meshes. The combination of h-refinement (adaptive meshing) and P refinement (higher-order methods) should also be further pursued since this has been shown to enable exponential convergence [12].

3. SOLUTION TECHNIQUES

Once the governing equations have been discretized in space, they form a large set of coupled ordinary differential equations which must be integrated in time. The main interest in this paper relates to the solution of steady-state problems. In this case, time accuracy of the integration may be sacrificed in the interest of accelerating the convergence to steady-state. This may include the use of a low-accuracy time integration scheme, the use of large time steps, and lumping of the mass matrix for finite-element schemes. Furthermore, many of the convergence acceleration techniques developed for structured meshes carry over in a straight forward manner to unstructured meshes. These include the use of local time stepping, enthalpy damping for inviscid flows, and implicit residual averaging [13,14] (which must be implemented using a Jacobi iteration rather than a tridiagonal solver).

However, for large problems, the use of simple explicit schemes inevitably results in very slow convergence rates. Many of the solution algorithms employed for structured grids exploit the structure of the grid (e.g ADI schemes) or the limited bandwidth of the resulting Jacobian matrix, and thus are not applicable to unstructured meshes. The lack of efficient steady-state solution algorithms for unstructured mesh problems has been one of the main impedements towards greater use of unstructured mesh strategies. For large stiff problems, implicit methods based on sparse matrix technology or multi-level approaches modified for use on unstructured data-sets are required.

The ultimate implicit method is the direct solver, or Newton's method. If an exact linearization of the Jacobian is employed, and the resulting matrix is inverted at each time-step using sparse matrix techniques, quadratic convergence can be obtained, resulting in convergence to machine zero in under ten iterations. Direct methods have been demonstrated for both structured grids and unstructured grids [15,16,17]. Although these are among the most robust methods available, their operation count and storage requirements for unstructured meshes are non-optimal and are thus seldom employed in practice.

For a second-order method, the exact linearization results in a stencil which includes nearest neighbors as well as second to nearest neighbors. Thus, the storage requirements for the resulting sparse matrix become excessive, particularly in three dimensions. By employing a linearization of the corresponding first-order scheme, a nearest neighbor stencil is obtained, and the memory requirements for storing the corresponding sparse matrix are reduced substantially. However, this mismatch between the implicit and explicit operators ensures that quadratic convergence rates cannot be achieved. Furthermore, the exact inversion of the implicit matrix at each time-step is no longer necessary, since the implicit matrix itself is an inaccurate representation of the explicit operator. Thus, the use of iterative implicit methods, in which the implicit system of equations is only approximately solved at each time-step is more appropriate.

A large variety of iterative implicit methods have been developed. These may consist of a single iteration scheme, or a preconditioning operation followed for example by a GMRES (generalized minimum residual) technique. These implicit methods may be divided into methods which operate pointwise (such as Jacobi and Gauss-Siedel methods), and those which require storing the entire implicit matrix (such as LU factorization schemes).

Jacobi and Gauss-Siedel approaches have been employed successfully both as iteration schemes and as preconditioners for GMRES [4,7,18,19,20]. Improved efficiency over explicit schemes with minimal memory overheads have been demonstrated for a variety of problems. However, for large problems and very stiff equation sets, such as those encountered in the solution of high Reynolds number viscous flows, the local nature of these methods results in a degradation of the convergence rate.

Methods which operate on the entire implicit matrix such as LU factorization are of a more global nature; such methods promote the rapid transmittal of information across the entire domain. As such, these methods are more robust and their convergence degrades less significantly for very large and stiff problems. An incomplete LU factorization employed as a preconditioner followed by a GMRES technique has been found to yield one of the most competitive solution strategies for high-Reynolds number viscous two-dimensional flows [4]. However, such methods require the storage of the entire implicit matrix. While this is feasible in two-dimensions, for three dimensional calculations this matrix alone requires of the order of 300 N words of storage for a vertex scheme, where N represents the number of mesh vertices.

An interesting alternative approach which is not entirely local, but which obviates the need to store the implicit Jacobian matrix is based on the use of additional data-structures called linelets or snakes [21,22]. By joining series of neighboring points together in the unstructured mesh, a set of lines can be identified and employed to mimick the alternating direction implicit (ADI) type algorithms commonly employed on structured meshes. Such methods may be viewed as a compromise between the low storage requirements of point-wise methods and the global nature of LU factorization methods, and thus their performance can be expected to fall somewhere in this region. However, for problems where the stiffness is strongly directional, such as for high Reynolds number boundary layers, this approach may be capable of resolving much of the stiffness.

An entirely different approach involves the use of multi-level or multigrid strategies. These are hierarchical methods which make use of a sequence of coarser grids to accelerate the solution on a fine grid. The advantages of time stepping on coarse meshes are twofold: first, the permissible time-step is much larger, since it is proportional to the cell size, and secondly, the work is much less because of the smaller number of grid points. Generally, a simple

explicit scheme is employed on each grid of the sequence. The process begins by performing a time-step on the finest grid of the sequence, and then interpolating the flow variables and residuals up to the next coarser grid of the sequence. On this grid, a correction equation is formulated, which consists of the governing flow equations augmented by a forcing function which represents the fine grid solution. This correction equation is time-stepped and the resulting flow variables and residuals are interpolated up to the next coarser grid, where the process is repeated recursively until the coarsest grid of the sequence is reached. The computed corrections are then recursively interpolated back down to the finest grid where they are employed to update the solution. This entire procedure constitutes one multigrid cycle. These cycles are repeated until convergence is obtained.

The use of a multigrid method with unstructured meshes presents an additional challenge. Consistent coarse tetrahedral grids can no longer be formed by simply grouping together neighboring sets of tetrahedra. An alternative would be to generate the fine mesh by repeatedly subdividing an initial coarse mesh in some manner. However, generally poor topological control of the fine mesh results from such a procedure. A strategy which has proven successful involves the use of independent non-nested coarse and fine grids. This approach provides great flexibility in determining the configuration of the coarsest and finest meshes. Coarse meshes may be designed to optimize the speed of convergence, whereas fine meshes may be constructed based on solution accuracy considerations. In general, beginning from a fine grid, a coarser level is constructed which contains roughly half the resolution in each coordinate direction throughout the domain (about 1/8 the number of vertices in three dimensions, or 1/4 in two dimensions). This process is repeated until the coarsest grid capable of representing the geometry topology is obtained. In the context of adaptive meshing, new finer meshes may be added to the multigrid sequence, using any given adaptive refinement technique, since no relation is assumed between the various meshes of the sequence. The key to the success of such a strategy lies in the ability to efficiently transfer variables, residuals and corrections back and forth between unrelated unstructured meshes. This may be performed using linear interpolation. For each vertex of a given grid, the tetrahedron which contains this vertex on the grid to which variables are to be interpolated is determined. The variable at this node is then linearly distributed to the four vertices of the enclosing tetrahedron (three vertices of the enclosing triangle in two dimensions). The main difficulty lies in efficiently determining the enclosing cell for each grid point. A naive search over all cells would lead to an $O(N^2)$ complexity algorithm, where N is the total number of grid points, and would be more expensive than the flow solution itself. Thus, an efficient search strategy such as a graph-traversal algorithm or a quad-tree approach is required.

This particular unstructured multigrid approach has been shown to be very effective [3,23,24,25]. Near grid independent convergence rates can be obtained while incurring minimal memory overheads. However, the need to manually generate a complete sequence of grids is viewed as tedious in a production environment, and several efforts at automating this process have been developed. On approach agglomerates or fuses together neighboring cells to form coarse super-cells which are generally not tetrahedral but polyhedral [26,27]. A second approach constructs triangular or tetrahedral grids by filtering a portion of the fine grid points and retriangulating the remaining points [28,29]. All of these approaches involve various tradeoffs. However, they all make use of extra geometrical constructs (i.e. coarse grids) to solve what is essentially an algebraic matrix inversion problem. This may be viewed as an inconvenience. However, unstructured multigrid methods are probably the most efficient

solution methods available presently in terms of CPU and memory overhead for steady-state solutions.

4. GRID GENERATION AND ADAPTIVITY

Although one of the main motivations for the use of unstructured meshes has been the added flexibility they offer for dealing with complex geometries, grid generation remains a pacing item for unstructured mesh computations, especially in three dimensions. To be sure, part of the problem is associated with the lack of standard and flexible geometry definition standards and interfaces to current CAD systems employed in the industrial design process. However, much difficulty still rests with the grid generation algorithms themselves.

Unstructured mesh generation algorithms have traditionally been divided into advancing front type methods, and Delaunay triangulation methods, although this classification is somewhat arbitrary. In fact, these two approaches are not mutually exclusive. The advancing front algorithm begins with a surface mesh on the geometry which it then marches out into the flow field by adding points ahead of the front and joining them up to form tetrahedra with existing front faces, until the entire region has been discretized [30,31]. This algorithm essentially represents a point placement strategy. The reconnection strategy employed to form tetrahedral elements is somewhat arbitrary and resorts to checking the validity of each proposed tetrahedral cell by examining possible intersections with neighboring cells. The advantages of such a method are that it guarantees the integrity of the boundaries. This is evident since the geometry surface-grid constitutes the original front, and the method is of the "greedy type"; i.e. it never undoes what has already been constructed. The resulting placement of the mesh points is generally very satisfactory and very smooth element variations can be ensured. On the other hand, robustness is not guaranteed, due to the somewhat heuristic nature of the reconnection strategy. Sophisticated dynamically varying data-structures are required to accelerate the spatial searching routines employed in such an approach.

A Delaunay triangulation represents one of the most fundamental data-structures in computational geometry [32]. Given a set of points in a three dimensional volume (or in a two dimensional plane), the Delaunay triangulation of these points constitutes a set of non-overlapping tetrahedra (or triangles in 2-D), the union of which define the convex hull of the points, and for which a number of properties can be proved. Various algorithms exist for constructing the Delaunay triangulation [33]. A commonly employed algorithm in the mesh generation context, known as Bowyer's [34] or Watson's [35] algorithm, is based on the empty circumsphere criterion. This property states that the circumsphere of any tetrahedron cannot contain any other vertices of the mesh. Thus, if an initial triangulation is assumed to exist, new mesh points can be introduced one at a time and triangulated into the mesh by first locating all tetrahedra whose circumsphere is intersected by the newly introduced point, removing all such elements, and forming new tetrahedra by joining the new point up to the faces of the cavity which was created by the removal of the intersected elements.

Since proofs exist for the validity of Delaunay triangulation algorithms, robustness can potentially be built into a mesh generator by making use of such algorithms. However, the Delaunay triangulation is only valid within the convex hull of its defining points. Thus for geometries other than the convex hull, such as a body inside a flowfield, the integrity of the geometry cannot be guaranteed. One approach to this problem is to triangulate the entire set of mesh points, and then to attempt to reconstruct a prescribed surface grid on the geometry by swapping edges and faces of the mesh [36]. Another approach ensures that the placement of

points in the mesh is such that the geometry integrity is observed [37].

Since a Delaunay triangulation merely describes a connectivity pattern for a set of points, arbitrary point placement strategies can be employed. In general, the point placement strategies employed are somewhat heuristic. Point sets may be pre-determined by sets of overlapping structured grids, or generated incrementally by subdividing elements deemed to be too large [38]. These strategies have generally resulted in less than optimal point distributions, and the resulting meshes are generally less smooth than those obtained with the advancing front method.

On the other hand, Delaunay triangulation mesh generation strategies based on Bowyer's algorithm have proved to be extremely efficient and rather simple to implement. They obviate the need for the complex data-structures required in the advancing front technique, and do not perform tedious intersection checking. Perhaps an additional reason for such efficiency is the fact that Bowyer type algorithms generate the mesh one point at a time, whereas advancing front type algorithms proceed one tetrahedron at a time, and a typical unstructured mesh contains 5 to 6 times more tetrahedra than vertices.

As mentioned previously, adaptive meshing represents one of the principal advantages of the use of unstructured meshes. Similarly to the original mesh generation process, mesh adaptation requires the introduction (or removal) of new points at appropriate locations, and the reconnection of these points to neighboring points of the mesh. The simplest implementation of adaptive mesh refinement is to subdivide existing tetrahedra into smaller cells by introducing new points midway along the tetrahedra forming edges, and reconnecting these points according to a predetermined set of rules. This strategy is very efficient and simple to implement but can lead to distorted cells and vertices of very high degree after numerous refinement levels.

Bowyer's algorithm for Delaunay triangulation is ideally suited for adaptive meshing. Assuming the flow has been solved on a mesh which constitutes a Delaunay triangulation, the solution can be examined to determine regions of high discretization errors or large flow gradients where additional grid points are required. Each new grid point is then inserted into the mesh and locally retriangulated using Bowyer's algorithm, thus resulting in the Delaunay triangulation of the new augmented point set.

The advancing front technique makes use of a field function for determining the desired size of the mesh elements throughout the flow field. A simple adaptive remeshing strategy consists of replacing this initial field function by a function derived from the computed flow solution, and completely regenerating the entire mesh. This approach contains more flexibility for generating the new adaptive mesh, but is more expensive since a non-local mesh restructuring is performed, and may be impractical for transient type problems where many remeshings are required. Alternatively, individual regions of the mesh can be cut out, thereby defining new fronts to be advanced, and local mesh patches regenerated. (For an implementation of this procedure for the removal of distorted elements see [39]).

The main issue which needs to be addressed for all mesh generation and adaptation strategies is that of robustness. This can only be achieved through less heuristic and more theoretically sound approaches to the problem. New developments in computational geometry should enable the formulation of fool-proof algorithms. For example, the existence and construction methods for a constrained Delaunay triangulation in an arbitrary non-convex two-dimensional domain are now well known [40]. A constrained Delaunay triangulation is one which contains certain predefined edges in the final triangulation. Thus, the triangulation of a given set of

points in an arbitrary two-dimensional domain with initial geometry boundaries is a relatively easily solved problem. What is required is the extension of such proofs to three dimensions, as well as solutions to various optimization problems, such as: what is the optimal distribution of points ? the optimal triangulation of these points (it need not necessarily be a Delaunay triangulation)? how does such a triangulation interact with the solver? and how can one construct such a triangulation in an efficient and rigorous manner?

These issues become even more important for the solution of high-Reynolds number viscous flows. In three dimensions, few if any practical high-Reynolds number viscous flow solutions have been demonstrated. In two-dimensions, turbulent viscous flow solution capabilities have only emerged in the last several years. Aside from the turbulence modeling issues, which will be described in the results section of this paper, the main difficulty for solving such flows on unstructured meshes relates to the requirement of generating very highly stretched triangular cells (or tetrahedral elements in three dimensions) which are required in order to efficiently resolve the thin shear layers which occur in such flows. This represents a somewhat non-standard application of unstructured meshes, since highly stretched triangular elements have traditionally been considered detrimental to numerical accuracy, and as such have been avoided. However, it has been shown that, while triangular elements with one large angle ($\sim$ 180 degrees) are detrimental, elements with small angles, such as a highly elongated right angle triangle are acceptable [41]. This interplay between numerical behavior and optimal triangulation (for a given numerical method) is important for accurately and efficiently resolving complex flows, and should increasingly result in a tighter coupling between the grid generation and flow solution processes.

The current method employed by the author for generating highly stretched two-dimensional triangulations consists of constructing a Delaunay triangulation of the set of grid points in a mapped space rather than in physical space [42,43]. The set of grid points is determined by generating a stretched structured grid over each individual component of the geometry and considering the union of the points defined by these overlapping grids. The locally mapped space is defined by the amount of local grid stretching desired, which in turn is dictated by the aspect ratios of the underlying structured grid cells. In this mapped space, the mesh points appear locally isotropic, and a regular Delaunay triangulation is constructed. The projection of this triangulation back into the physical space produces the desired stretching. Although this method does not guarantee the formation of non-obtuse triangles, the combination of the particular point distribution and reconnection strategy tends to produce nearly right-angle triangles in regions of high stretching.

Improvements to this strategy can be sought by drawing on more rigorous computational geometry algorithms which provide bounds on the angles of the generated triangles, or combined with point placement optimization techniques. These techniques will become necessary for the generation of suitable stretched meshes in three dimensions for viscous flow calculations.

Another alternative to the generation of highly stretched triangular or tetrahedral meshes for the resolution of viscous flows is the use of hybrid meshes, where a thin layer of quadrilaterals in two-dimensions, or prizms in three-dimensions [44,45] are employed in the boundary layer regions. These methods by-pass the difficulties associated with the generation and solution of flows on highly stretched triangular and tetrahedral elements. On the other hand, the resulting grid becomes structured in one of the directions, and part of the generality of the unstructured approach is lost. Such trade-offs must of course be weighed in terms of the

complexity of the geometry.

5. TWO-DIMENSIONAL RESULTS

5.1. An Inviscid Case

In order to demonstrate the potential effectiveness of an unstructured mesh strategy, the solution of a steady-state inviscid internal flow, which incorporates adaptive meshing in conjunction with an unstructured multigrid algorithm is demonstrated. The basic discretization employed consists of a Galerkin finite-element approach with added artificial dissipation. The unstructured multigrid scheme makes use of a sequence of pre-generated unrelated coarse meshes, and mesh adaptation is achieved by introducing new points in regions of high density gradients and restructuring the mesh locally using Bowyer's algorithm.

The geometry consists of a two-dimensional turbine blade cascade which has been the subject of an experimental and computational investigation at the occasion of a VKI lecture series [46]. A total of seven meshes were used in the multigrid algorithm, with the last three meshes generated adaptively. The coarsest mesh of the sequence contains only 51 points, while the finest mesh, depicted in Figure 1, contains 9362 points. Extensive mesh refinement can be seen to occur in the neighborhood of shocks, and in other regions of high gradients. The inlet flow incidence is 30 degrees, and the average inlet Mach number is 0.27. The flow is turned 96 degrees by the blades, and the average exit isentropic Mach number is 1.3. At these conditions, the flow becomes supersonic as it passes through the cascade, and a complex oblique shock wave pattern is formed. These are evident from the computed Mach contours depicted in Figure 2. All shocks are well resolved, including some of the weaker reflected shocks, which non-adapted mesh computations often have difficulty resolving. Details of the flow in the rounded trailing edge region of the blade, where the flow separates inviscidly and forms a small recirculation region, are also well reproduced. Once the first four globally generated meshes were constructed, the entire flow solution - adaptive mesh enrichment cycle was performed three times, executing 25 multigrid cycles at each stage. This entire operation required 40 CPU seconds on a single processor of a Cray-YMP supercomputer. The residuals on the finest mesh were reduced by two and a half orders of magnitude, which should be adequate for engineering calculations.

5.2. Viscous Flows

The main difficulties involved in computing high-Reynolds-number viscous flows relate to the grid generation and turbulence modeling requirements. Since the grid generation issues have been previously discussed, the turbulence modeling issues will be briefly addressed in this section.

The most common turbulence models employed for aerodynamic flows are of the algebraic type. Such models typically require information concerning the distance of each point from the wall. Turbulence length scales are determined by scanning appropriate flow variables along specified streamwise stations. In the context of unstructured meshes, such information is not readily available and hence, the implementation of algebraic turbulence models on such meshes introduces additional complexities. A particular approach adopted by the author [47] consists of generating a set of background turbulence mesh stations. These are constructed by generating a hyperbolic structured mesh about each geometry component, based on the boundary-point distribution of the original unstructured mesh, and extracting the normal lines

of the mesh. When performing adaptive meshing, new turbulence mesh stations must be constructed for each new adaptively generated boundary point, as illustrated in Figure 3. Each time the turbulence model is executed, the flow variables are interpolated onto the normal turbulence stations, the turbulence model is executed on each station, and the resulting eddy viscosity is interpolated back to the unstructured mesh. The method employed for interpolating variables back and forth between the unstructured mesh and the turbulence mesh stations is similar to that previously described for the unstructured multigrid algorithm.

Figures 4 through 7 illustrate a calculation which makes use of these various techniques to compute a complicated two-dimensional viscous flow over a high-lift multi-element airfoil. The final mesh employed is depicted in Figure 4, and contains a total of 48,691 points. This mesh was obtained using the stretched Delaunay triangulation technique previously described, followed by two levels of adaptive refinement. The height of the smallest cells at the wall is of the order of 2×10^{-5} chords and cell aspect ratios up to 500:1 are observed. The computed Mach number contours for this case are depicted in Figure 5. The freestream Mach number is 0.1995, the chord Reynolds number is 1.187 million, and the corrected incidence is 16.02 degrees. At these conditions, the flow remains entirely subcritical. Compressibility effects are nevertheless important due to the large suction peaks generated about each airfoil. For example, in the suction peak on the upper surface of the leading-edge slat, the local Mach number achieves a value of 0.77. The computed surface pressure coefficients are compared with experimental wind tunnel data [48] in Figure 6, and good overall agreement is observed, including the prediction of the height of the suction peaks. This case provides a good illustration of the importance of adaptive meshing in practical aerodynamic calculations. Adequate resolution of the strong suction peak on the upper surface of the slat can only be achieved with a very fine mesh resolution in this region. Failure to adequately capture this large suction peak results in the generation of numerical entropy which is then convected downstream, thus contaminating the solution in the downstream regions, and degenerating the global accuracy of the solution. Because these suction peaks are very localized, they are efficiently resolved with adaptive techniques. In order to obtain a similar resolution using global mesh refinement, of the order of 200,000 mesh points would be required, greatly increasing the cost of the computation. The convergence history for this case, as measured by the density residuals and the total lift coefficient versus the number of multigrid cycles, is depicted in Figure 7. A total of 400 multigrid cycles were executed, which required roughly 35 minutes of single processor CRAY-YMP time, and 14 Mwords of memory.

The discrepancy between the computed and experimental pressure coefficients on the trailing edge flap is due to a separated flow condition which is not reproduced by the algebraic turbulence model. These results strongly indicate the need for more sophisticated turbulence modeling. The use of single or multiple field-equation models appears to be the most appropriate choice for turbulent unstructured mesh computations. Such models can be discretized in a straight-forward manner on unstructured meshes. However, the task is now to ensure that such models adequately represent the flow physics, and that they can be solved in an efficient and robust manner. The implementation of a standard high-Reynolds-number $k - \varepsilon$ turbulence model with low-Reynolds-number modifications proposed by Speziale, Abid and Anderson [49], is demonstrated in the next example. The main effort was focused on devising a technique for efficiently solving the two turbulence equations in the context of the unstructured multigrid strategy [50]. The four flow equations and the two turbulence equations are solved as a loosely coupled system. The flow equations are solved explicitly, and the turbulence

equations point-implicitly, using a time-step limit which ensures stability and positivity of k and ε. In the context of the unstructured multigrid algorithm, the turbulence eddy viscosity is assumed constant on all but the finest grid level where it is recomputed at each time-step. The transonic flow over a two-element airfoil configuration has been computed using this implementation of the model. For this case, the freestream Mach number is 0.5, the incidence is 7.5 degrees, and the Reynolds number is 4.5 million. Figures 8 and 9 depict the mesh and the solution obtained with the $k - \varepsilon$ turbulence model. Four meshes were employed in the multigrid sequence, with the finest mesh containing a total of 28,871 points. The convergence rates of the various equations for this case are plotted in Figure 10. As can be seen, the turbulence equations and flow equations converge at approximately the same rates. All flow variables and turbulence quantities are initialized with freestream values, and convergence to steady-state is achieved in several hundred multigrid cycles. The computed flow field exhibits regions of transonic flow with a small region of separated flow at the foot of the shock. These features are well reproduced by the turbulence model. Future efforts should concentrate on computationally predicting flows with large regions of separation, such as that inferred by Figure 6, and on developing models which better represent the flow physics.

6. THREE DIMENSIONAL RESULTS

Due to the limitations of present day supercomputers, and the difficulties associated with generating highly stretched tetrahedral meshes, three-dimensional computations have generally been confined to inviscid flows. Most of the techniques described in the context of two-dimensional inviscid flows extend readily to three dimensions. In particular, the unstructured multigrid algorithm and the adaptive meshing strategy have been found to be particular effective for three-dimensional computations [23]. As an example, an adaptive multigrid calculation of transonic flow about an ONERA M6 wing is illustrated in Figures 11 through 13. The final mesh, depicted in Figure 11, contains a total of 174,412 points and just over 1 million tetrahedral volumes. This represents the fourth mesh in the multigrid sequence and the second adaptive refinement level. Mesh refinement was based on the undivided gradient of density. The freestream Mach number and incidence for this case are 0.84 and 3.06 degrees respectively. The well known double shock pattern for this case is reproduced in the computed Mach contours of the solution in Figure 12. The leading edge expansion and shocks are well resolved due to the extensive mesh refinement in these regions. A globally refined mesh of this resolution would result in roughly 600,000 points and would thus require 3 to 4 times more computational resources. The multigrid convergence rate for this case is depicted in Figure 13, where 50 cycles were performed on the original grid, prior to adaptation, 50 cycles on the first adapted mesh, and 100 cycles on the finest adapted mesh. On this final mesh, the residuals were reduced by 5 orders of magnitude over 100 cycles, requiring a total of 35 CRAY-YMP single CPU minutes and 22 MW of memory.

6.1. Parallel Computing Results

As mentioned previously, due to their homogeneous (although random) nature, unstructured mesh data-sets are particularly well suited for parallel processing. An unstructured mesh solver typically consists of a single (indirect addressed) loop over all interior mesh edges, and another similar loop over all boundary elements. On a vector machine, each loop may be split into groups (colors) such that within each group, no recurrences occur. Each group can then be vectorized. A simple parallelization strategy for a shared memory machine is to further split

each group into n subgroups, where n is the number of available processors. Each subgroup can then be vectorized and run in parallel on its associated processor. Because the original number of groups is not large (usually 20 to 30), the vector lengths within each subgroup are still long enough to obtain the full vector speedup of the machine, for a moderate number of processors. For more massively parallel distributed-memory scalar machines, the entire mesh must be subdivided and each resulting partition associated with a single processor. On each processor, the single scalar interior and boundary loops are then executed, with inter-processor communication occurring at the beginning and end of each loop. The mesh partitioning strategy must ensure good load balancing on all processors while minimizing the amount of inter-processor communication required.

6.2. CRAY-YMP-8 Results

Figure 14 illustrates an unstructured mesh generated over a three-dimensional aircraft configuration. This mesh contains a total of 106,064 points and 575,986 tetrahedra. This represents the second finest mesh employed in the multigrid sequence. The finest mesh, which is not shown due to printing resolution limitations, contains a total of 804,056 points and approximately 4.5 million tetrahedra. This is believed to be the largest unstructured grid problem attempted to date. The inviscid flow was solved on this mesh using all eight processors running in parallel on the CRAY-YMP supercomputer. A total of 4 meshes were used in the multigrid sequence. The convergence rate for this case is depicted in Figure 16. In 100 multigrid cycles, the residuals were reduced by almost 6 orders of magnitude. This run required a total of 16 minutes wall clock time running in dedicated mode on the 8 processor CRAY-YMP, including the time to read in all the grid files, write out the solution, and monitor the convergence by summing and printing out the average residual throughout the flow field at each multigrid cycle. The total memory requirements for this job were 94 million words. The ratio of CPU time to wall clock time was 7.7 on 8 processors, and the average speed of calculation was 750 Mflops, as measured by the CRAY hardware performance monitor [51]. For this case, the freestream Mach number is 0.768 and the incidence is 1.116 degrees. The computed Mach contours are shown in Figure 15, where good resolution of the shock on the wing is observed, due to the large number of mesh points employed.

6.3. Intel Touchstone Delta Results

The implementation of the unstructured multigrid Euler solver on the Intel Touchstone Delta distributed memory scalar multiprocessor machine, has been pursued using a set of software primitives designed to ease the porting of scientific codes to parallel machines [52]. The present implementation was undertaken as part of a more general project aimed at designing and constructing such primitives with experience gained from various implementations. The net effect of the use of such primitives is to relieve the programmer of most of the low level machine dependent software programming tasks. The mesh was partitioned using a spectral partioning algorithm which had previously been shown to produce good load balancing and minimize inter-processor communication [53]. The flow over the aircraft configuration previously described using the 804,056 vertex mesh was recomputed on the Intel Touchstone Delta machine using both an explicit single grid unstructured euler solver, and the unstructured multigrid euler solver. The single grid solver achieved a computational rate of 1.5 gigaflops on 512 processors, whereas the multigrid solver, using a V-cycle strategy achieved a rate of 1.2 gigaflops on the same number of processors. This represents a computational efficiency of

50% to 60%. These numbers are based on a single processor speed of approximately 5 Mflops, which corresponds to the computational rate achieved for a series of small meshes which were run on a single processor. The computational efficiencies are seen to vary with the particular solution strategy employed, and were also observed to vary with the size of the mesh. On the other hand, the CRAY-YMP-8 results were found to be relatively insensitive to the solution algorithm or the problem size. This is presumably due to the large bandwidth and shared-memory architecture of the machine. However, on the Intel Touchstone Delta, the multigrid strategy is still the method of choice, since in spite of its slightly lower computational efficiency, the numerical efficiency (convergence rate) achieved by this approach is approximately an order of magnitude greater than that of the simple single-grid explicit scheme. For this case, convergence to steady-state could be achieved in approximately 10 minutes of wall clock time using 512 processors.

7. CONCLUSION

This paper has illustrated the application of unstructured mesh techniques to various types of aerodynamic flows, and emphasized the advantages which can be obtained for complex geometries using adaptive meshing and parallelization. In two dimensions, a viscous flow solution capability has been demonstrated, while in three dimensions, efficient Euler solutions are possible. The main problems associated with three-dimensional viscous solutions are related to the development of reliable grid generation strategies, particularly with regards to the generation of highly stretched tetrahedral elements for capturing thin viscous layers. Turbulence modeling is also a limiting factor, although this difficulty is not particular to the field of unstructured meshes. Future work should also concentrate on more complete parallelization of the entire solution process, including items such as grid generation, partitioning, and adaptive meshing.

ACKNOWLEDGEMENTS

Many of the results shown here have been made possible due to the generous assistance of a large number of people. Among those, Daryl Bonhaus is acknowledged for his work in helping validate the 2-D viscous flow solver. The help of Shahyar Pirzadeh and Clyde Gumbert in providing assistance with the generation of three-dimensional unstructured grids using the VGRID program is acknowledged. Rob Vermeland and CRAY Research Inc. are thanked for providing dedicated time on the CRAY-YMP-8 machine, and the National Aerodynamic Simulation Facility (NAS), for providing the computational facilities which have made most of this work possible. Finally, Raja Das, Joel Saltz and Ravi Ponnusamy are acknowledged for actually having done most of the work on the distributed memory parallel implementation.

REFERENCES

1. Dannenhoffer, J. F., "Grid Adaptation for Complex Two-Dimensional Transonic Flows", *Sc.D Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology*, August 1987.

2. Jameson, A., Baker, T. J., and Weatherill, N. P., "Calculation of Inviscid Transonic Flow over a Complete Aircraft", *AIAA paper 86-0103*, January, 1986.

3. Mavriplis, D. J., "Turbulent Flow Calculations Using Unstructured and Adaptive Meshes", *Int. J. Numer. Methods Fluids, Vol. 13, No. 9*, November 1991, pp. 1131-1152

4. Venkatakrishnan, V., and Mavriplis, D., "Implicit Solvers for Unstructured Meshes", *AIAA paper 91-1537CP, Proceedings of the 10th AIAA Computational Fluid Dynamics Conference,* Honolulu, Hawaii, June 1991

5. Turkel, E., and Vatsa, V. N., "Effect of Artificial Viscosity on Three Dimensional Flow Solutions", *AIAA paper 90-1444* June 1990.

6. Stoufflet, B., Periaux, J., Fezoui, F., and Dervieux, A., "Numerical Simulation of 3-D Hypersonic Euler Flows Around Space Vehicles Using Adapted Finite Elements", *AIAA paper 87-0560* January 1987.

7. Batina, J. T., "Implicit Flux-Split Euler Schemes for Unsteady Aerodynamic Analysis Involving Unstructured Dynamic Meshes", *AIAA paper 90-0936,* April 1990.

8. Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes" *AIAA paper 89-0366* January, 1989.

9. Struijs, R., Deconinck, H., de Palma, P., Roe, P., and Powell, K. G., "Progress on Multidimensional Upwind Euler Solvers for Unstructured Grids", *AIAA paper 91-1550CP, Proceedings of the 10th AIAA Computational Fluid Dynamics Conference,* Honolulu, Hawaii, June 1991

10. Barth, T. J. and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction", *AIAA paper 90-0013,* January 1990.

11. Halt, D. W., and Agarwal, R. K., "A Compact Higher-Order Euler Solver for Unstructured Grids with Curved Boundaries" *AIAA paper 92-2696,* June 1992.

12. Oden, J. T., Demkowicz, L., Liszka, T., and Rachowicz, W., "h-p Adaptive Finite Element Methods for Compressible and Incompressible Flows", *Proceedings of the Symposium on Computational Technology on Flight Vehicles, Eds. A. K. Noor, S. L. Venneri, Permagon Press, pp. 523-534.* Washington, D.C., November 1990

13. Jameson, A., "Transonic Flow Calculations" *Princeton University Report MAE 1751,* 1984

14. Mavriplis, D. J., and Jameson, A., "Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes", *AIAA Journal, Vol 28, No. 8, pp. 1415-1425,* August 1990.

15. Venkatakrishnan, V., "Newton Solution of Inviscid and Viscous Problems", *AIAA paper 88-0413,* January 1988.

16. Orkwis, P. D., and McRae, D. S., "A Newton's Method Solver for the Navier Stokes Equations", *AIAA paper 90-1524,* June 1990.

17. Venkatakrishnan, V. and Barth, T.J., "Application of Direct Solvers to Unstructured Meshes for the Euler and Navier-Stokes Equations Using Upwind Schemes", *AIAA Paper 89-0364,* January, 1989.

18. Struijs, R., Vankeirsbilck, P., Deconinck, H., "An Adaptive Grid Polygonal Finite-Volume Method for the Compressible Flow Equations", *AIAA paper 89-1959CP* June 1989.

19. Fezoui, F., Stoufflet, B., "A Class of Implicit Upwind Schemes for Euler Simulations with Unstructured Meshes", *Journal of Computational Physics,* Vol 84, No. 1, September 1989, pp. 174-206.

20. Whitaker, D. L., Slack, D. C., Walters, R. W., "Solution Algorithms for the Two-Dimensional Euler Equations on Unstructured Meshes", *AIAA paper 90-0697* January 1990.

21. Hassan, O. Morgan, K., Peraire, J., "An Implicit Finite-Element Method for High Speed Flows", *AIAA paper 90-0402,* January 1990.

22. Lohner, R., Martin, D., "An Implicit Linelet-Based Solver for Incompressible Flows", *AIAA paper 92-0668* January 1992.

23. Mavriplis, D. J., "Three Dimensional Unstructured Multigrid for the Euler Equations", *Proc. of the AIAA 10th Comp. Fluid Dyn. Conf.,* AIAA paper 91-1549, June, 1991.

24. Leclercq, M. P., "Resolution des Equations d'Euler par des Methodes Multigrilles Conditions aux Limites en Regime Hypersonique", *Ph.D Thesis, Applied Math, Universite´de Saint-Etienne,* April, 1990.

25. Peraire, J., Peiro, J., and Morgan K., "A 3D Finite-Element Multigrid Solver for the Euler Equations" *AIAA paper 92-0449* January 1992.

26. Lallemand, M. H., Dervieux, A., A Multigrid Finite-Element Method for Solving the Two-Dimensional Euler Equations", *Proceedings of the Third Copper Mountain Conference on Multigrid Methods, Lecture Notes in Pure and Applied Mathematics,* Ed S. F. McCormick, Marcel Dekker Inc., April 1987, pp. 337-363.

27. Smith, W. A., "Multigrid Solution of Transonic Flow on Unstructured Grids", *Recent Advances and Applications in Computational Fluid Dynamics, Proceedings of the ASME Winter Annual Meeting,* Ed. O. Baysal, November 1990.

28. Morano, E., Guillard, H., Dervieux, A., Leclercq, M. P., Stoufflet, B., "Faster Relaxations for Non-Structured MG with Voronoi Coarsening", *Proceedings of the First European Computational Fluid Dynamics Conference, Eds. Ch. Hirsch,* Brussels, September 1992.

29. Vassberg, J., "A Fast Implicit Unstructured Mesh Euler Method", *AIAA paper 92-2693,* June 1992.

30. Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C., "Adaptive Remeshing for Compressible Flow Computations", *J. Comp. Phys.,* Vol 72, October, 1987, pp. 449-466

31. Parikh, P., Pirzadeh, S., and Lohner, R., "A Package for 3-D Unstructured Grid Generation, Finite-Element Flow Solution and Flow-Field Visualization" *NASA CR-182090* September 1990.

32. Aurenhammer, F., "Voronoi Diagrams - A survey of a Fundamental Geometric Data Structure" *ACM Comput. Surveys,* Vol 23, 1991, pp. 345-406

33. Preparata, F. P., and Shamos, M. I., *Computational Geometry, An Introduction,* Texts and Monographs in Computer Science, Springer-Verlag, 1985.

34. Bowyer, A., "Computing Dirichlet Tessalations", *The Computer Journal,* Vol. 24, No. 2, 1981, pp. 162-166

35. Watson, D. F., "Computing the n-dimensional Delaunay Tessalation with Application to Voronoi Polytopes", *The Computer Journal, Vol 24, No. 2,* pp. 167-172, 1981.

36. George, P. L., Hecht, F., and Saltel, E., "Fully Automatic Mesh Generator for 3D Domains of any Shape", *Impact of Computing in Science and Engineering, Vol 2, No. 3,* pp. 187-218, 1990.

37. Baker, T. J., "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", *Proc. of the AIAA 8th Comp. Fluid Dyn. Conf.,* AIAA paper 87-1124, June, 1987.

38. Holmes, D. G., and Snyder, D. D., "The Generation of Unstructured Meshes Using Delaunay Triangulation" Numerical Grid Generation in Computational Fluid Mechanics *Proc. of the Second International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Miami, December 1988,* Eds. S. Sengupta, J. Hauser, P. R.

Eisman, and J. F. Thompson, Pineridge Press Ltd., 1988.

39. Pirzadeh, S., "Recent Progress in Unstructured Grid Generation", *AIAA paper 92-0445* January, 1992.

40. Chew, L. P., "Constrained Delaunay Triangulations", *Algorithmica, Vol 4,* pp. 97-108, 1989.

41. Babuska, I., and Aziz, A. K., "On the Angle Condition in the Finite Element Method", *SIAM Journal of Numerical Analysis,* Vol 13, No. 2, 1976.

42. Mavriplis, D. J., "Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation" *Journal of Comp. Physics, Vol. 90, No. 2,*

43. Mavriplis, D. J., "Unstructured and Adaptive Mesh Generation for High-Reynolds Number Viscous Flows", *Proc. of the Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields,* Eds. A. S. Arcilla, J. Hauser, P. R. Eiseman, J. F. Thompson, North Holland, June 1991.

44. Nakahashi, N., "FDM-FEM Zonal Approach for Viscous Flow Computations Over Multiple Bodies", *AIAA paper 87-0604,* January, 1987.

45. Kallinderis, Y., and Ward, S., "Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations", *AIAA paper 92-2721* June 1992.

46. Sieverding, C. H., "Experimental Data on Two Transonic Turbine Blade Sections and Comparisons with Various Theoretical Methods", *Transonic Flows in Turbomachinery,* VKI Lecture Series 59, 1973.

47. Mavriplis, D. J., "Algebraic Turbulence Modeling for Unstructured and Adaptive Meshes", *AIAA Paper 90-1653,* June, 1990.

48. Wigton, L. B., *Private Communication,* The Boeing Company

49. Speziale, C. G., Abid, R., and Anderson, E. C., "A Critical Evaluation of Two-Equation Models for Near Wall Turbulence", *ICASE Report 90-46, NASA CR 182068, AIAA paper 90-1481,* June, 1990

50. Mavriplis, D. J., "Multigrid Solution of Compressible Turbulent Flow on Unstructured Meshes Using a Two-Equation Model", *AIAA Paper 91-0237,* January 1991.

51. UNICOS Performance Utilities Reference Manual, *SR-2040 6.0* Cray Research Inc.

52. Das, R., Mavriplis, D. J., Saltz, J., Ponnusamy, R., "The Design and Implementation of a Parallel Unstructured Euler Solver Using Software Primitives", *AIAA paper 92-0562,* January 1992.

53. Simon, H., "Partitioning of Unstructured Mesh Problems for Parallel Processing", *Proceedings of the Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications,* Permagon Press, 1991.

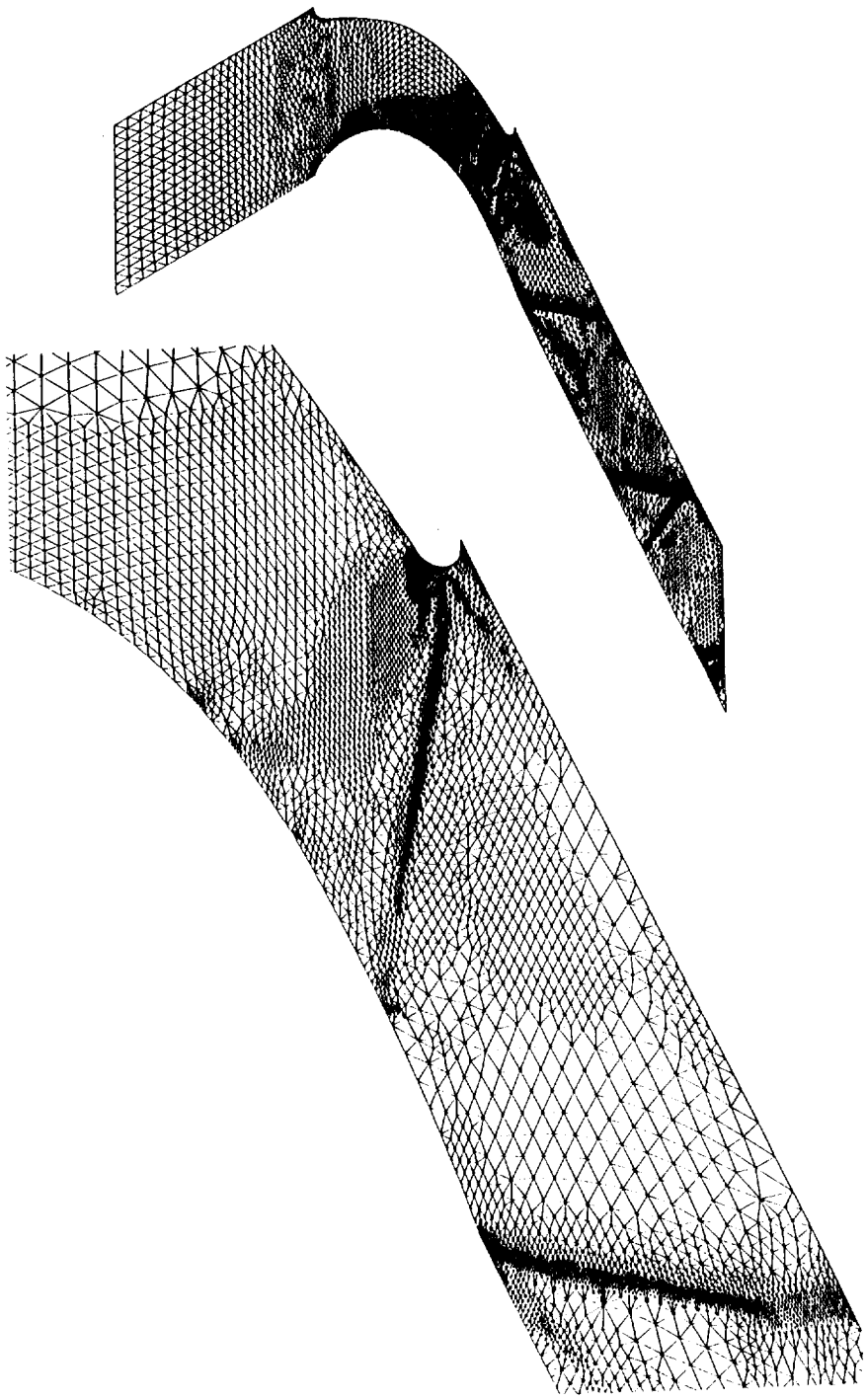

Figure 1
Adaptive Mesh Employed for Computing Transonic Inviscid Flow Through a Periodic Turbine Blade Cascade Geometry; Number of Nodes = 9362

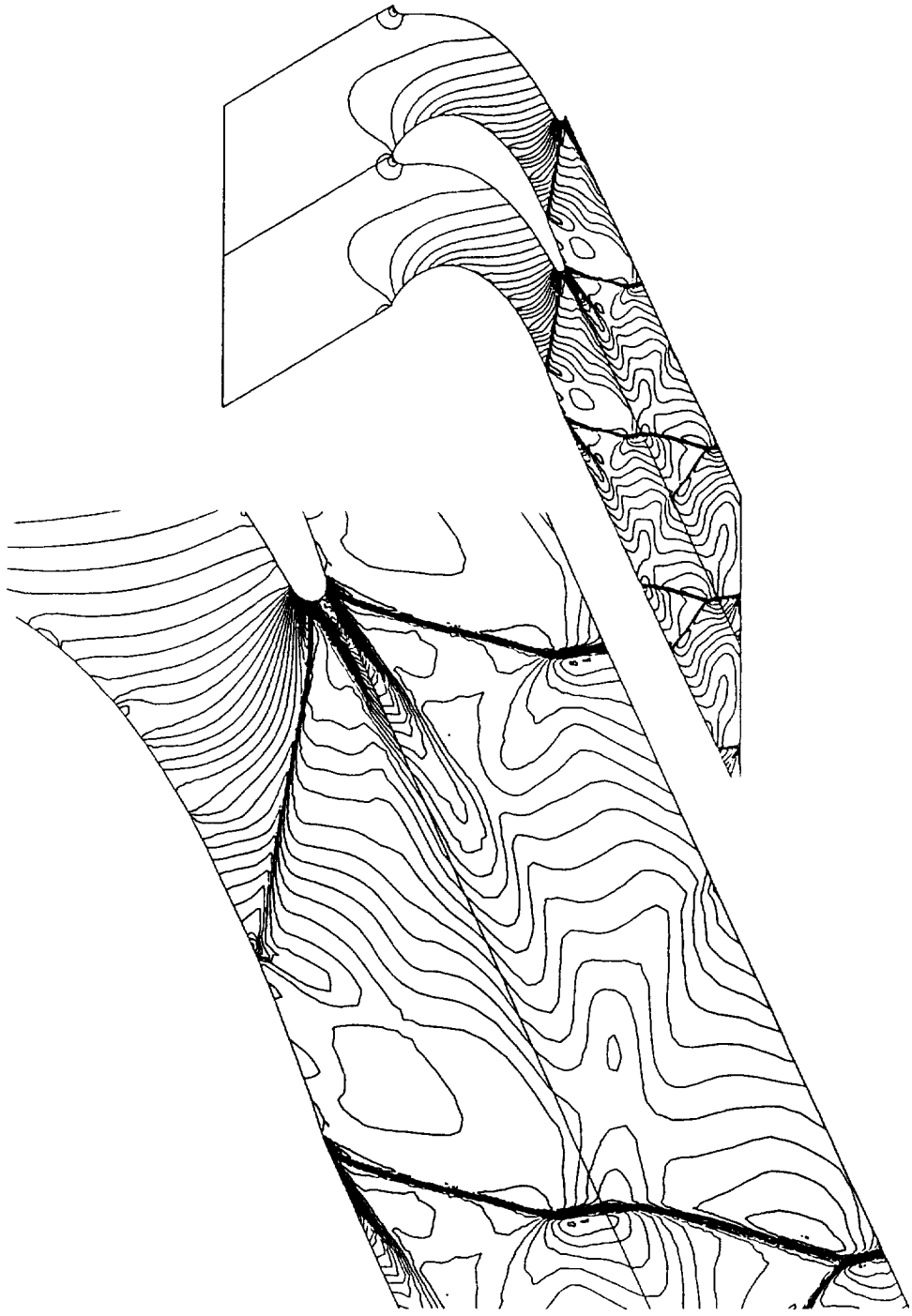

Figure 2
Computed Mach Contours for Flow Through a Periodic Turbine Blade Cascade Geometry

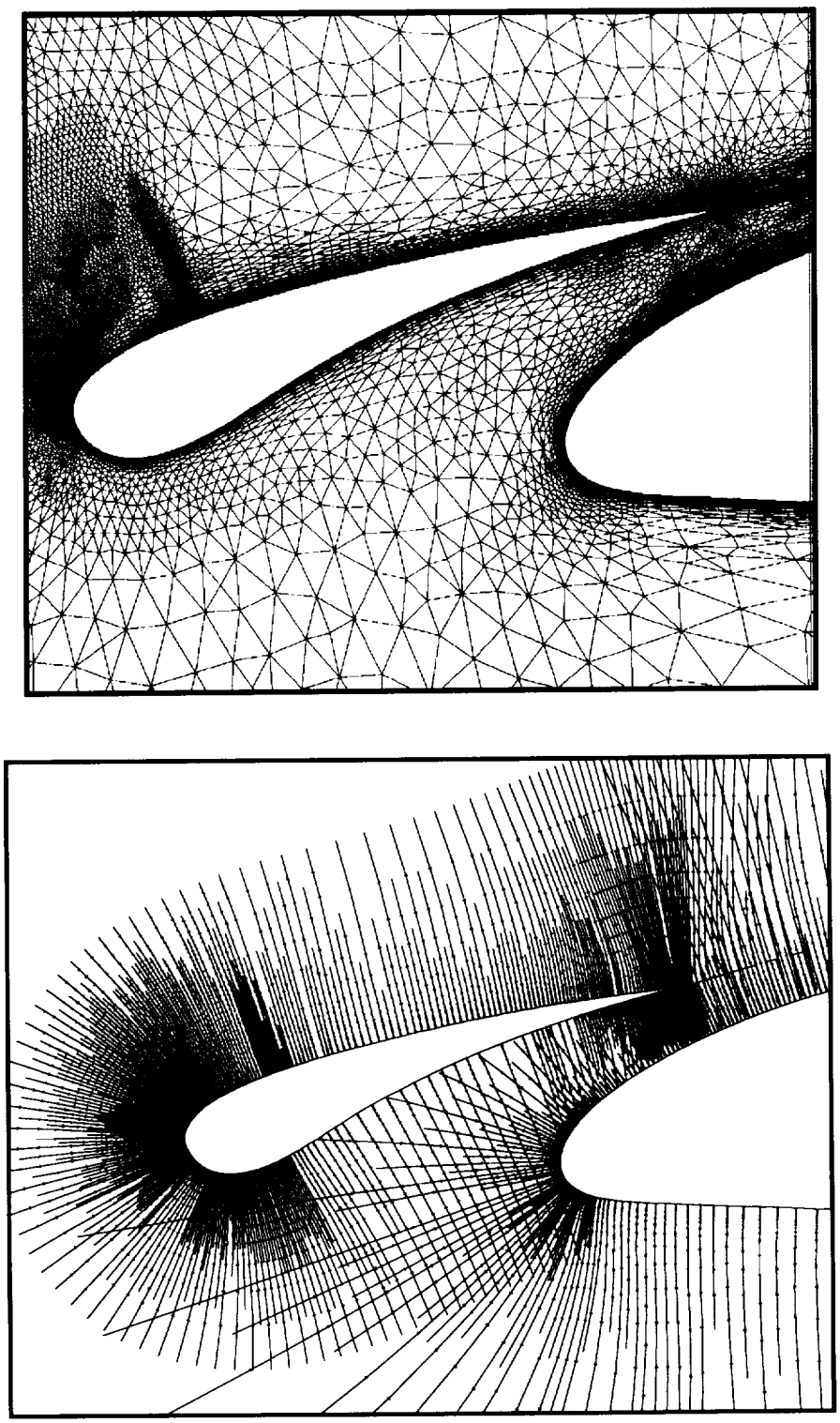

Figure 3
Illustration of Turbulence Mesh Stations Employed in Algebraic Model
for an Adaptively Generated Mesh

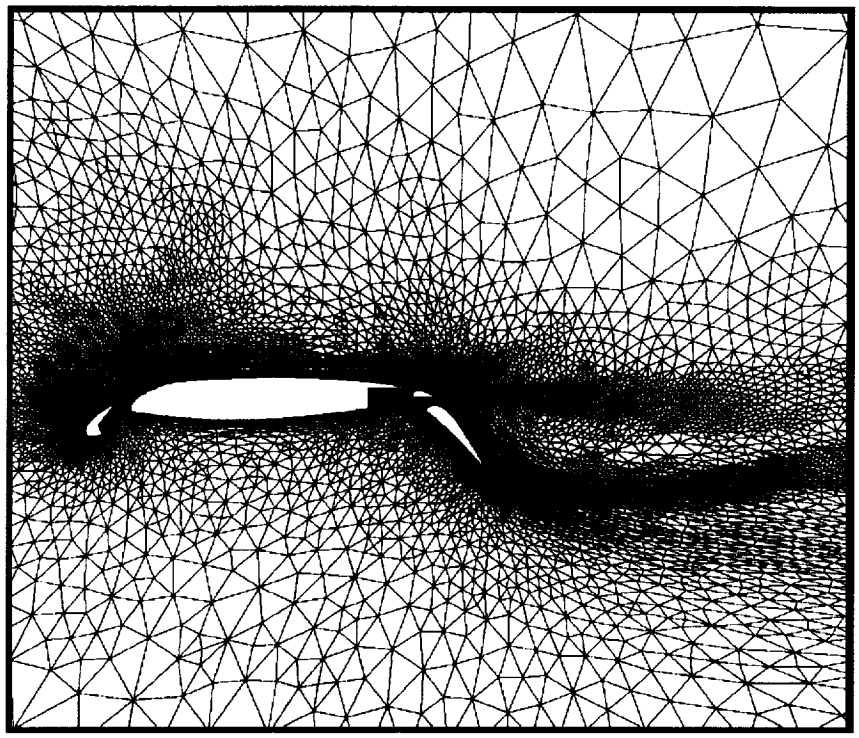

Figure 4
Adaptively Generated Unstructured Mesh about Four-Element Airfoil;
Number of Nodes = 48,691

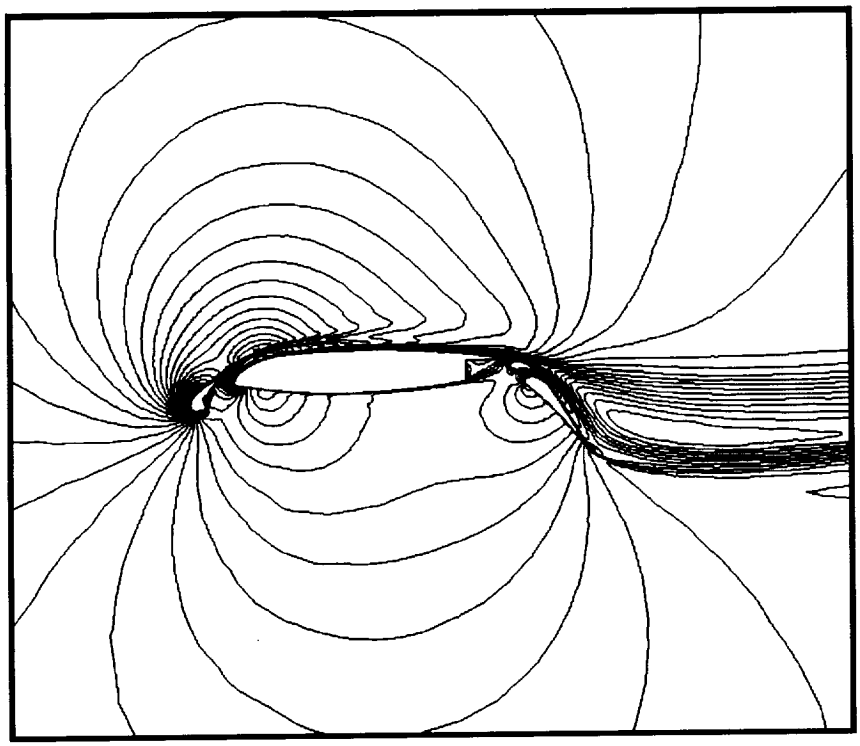

Figure 5
Computed Mach Contours for Flow over Four-Element Airfoil;
Mach = 0.1995, Reynolds Number = 1.187 million, Incidence = 16.02 degrees

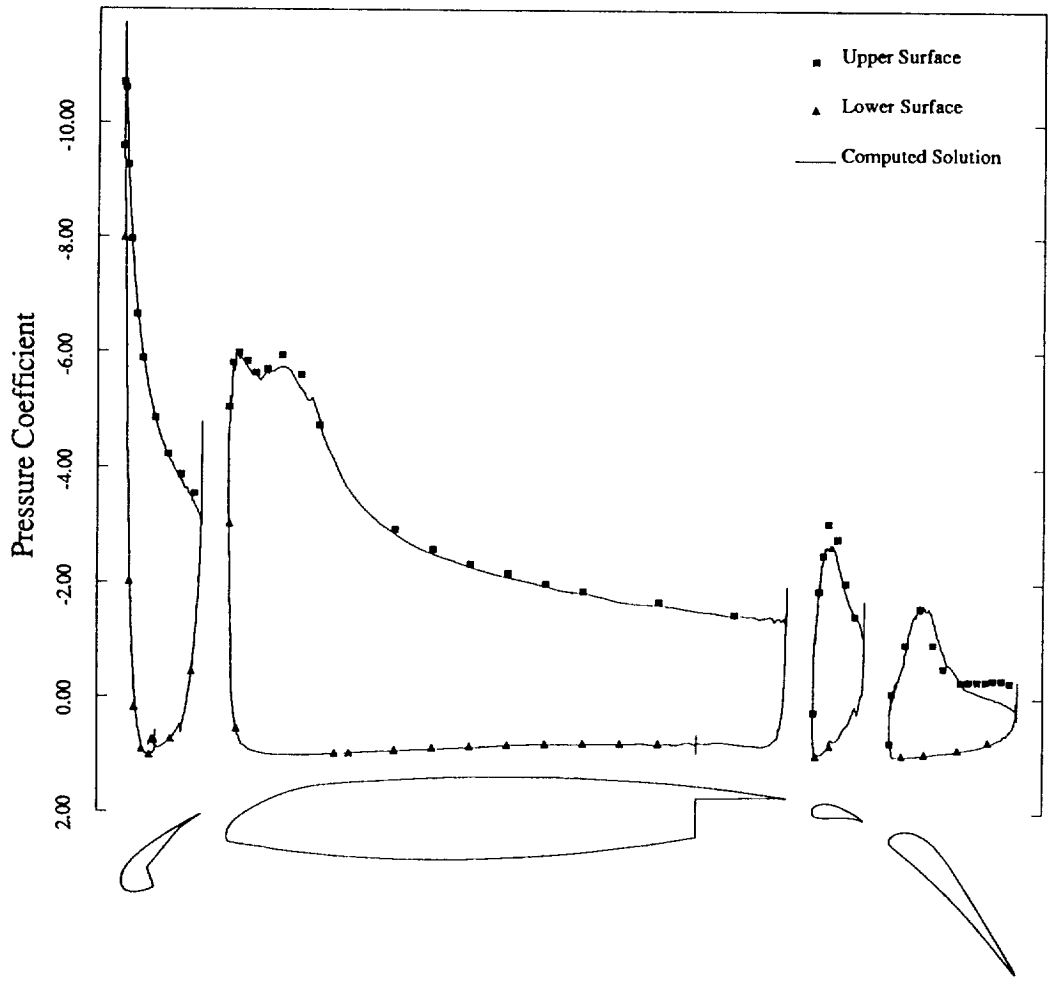

Figure 6
Comparison of Computed Surface Pressure Distribution with Experimental Wind-Tunnel Data for Flow Over Four-Element Airfoil Configuration; Mach = 0.1995, Reynolds Number = 1.187 million, Incidence = 16.02 degrees

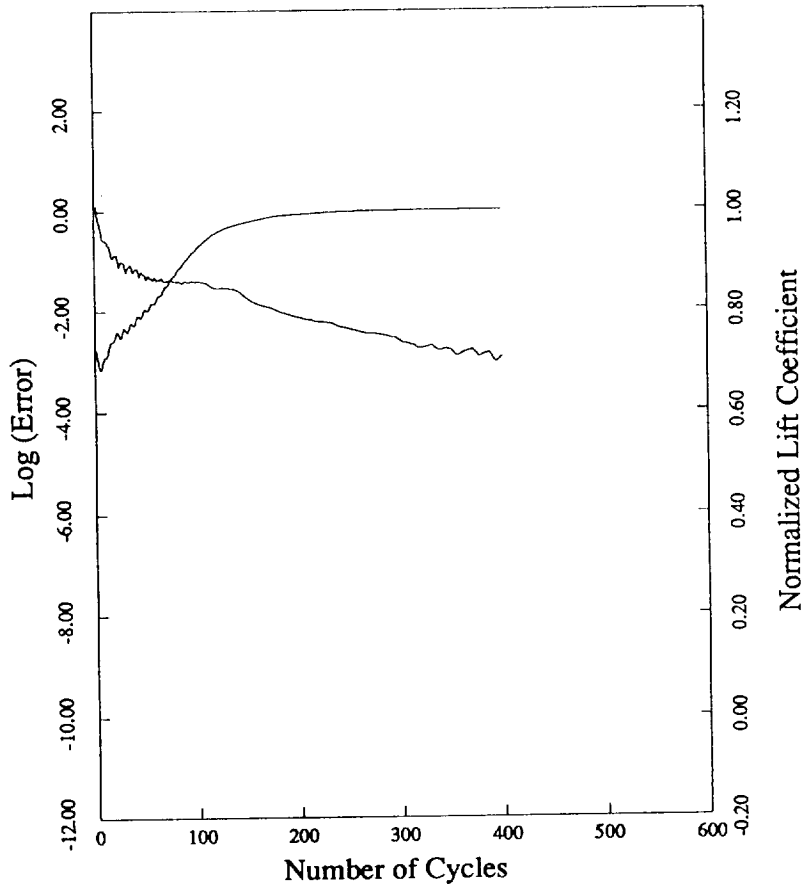

Figure 7
Convergence as Measured by the Computed Lift Coefficient and the Density Residuals Versus the Number of Multigrid Cycles for Flow Past a Four-Element Airfoil

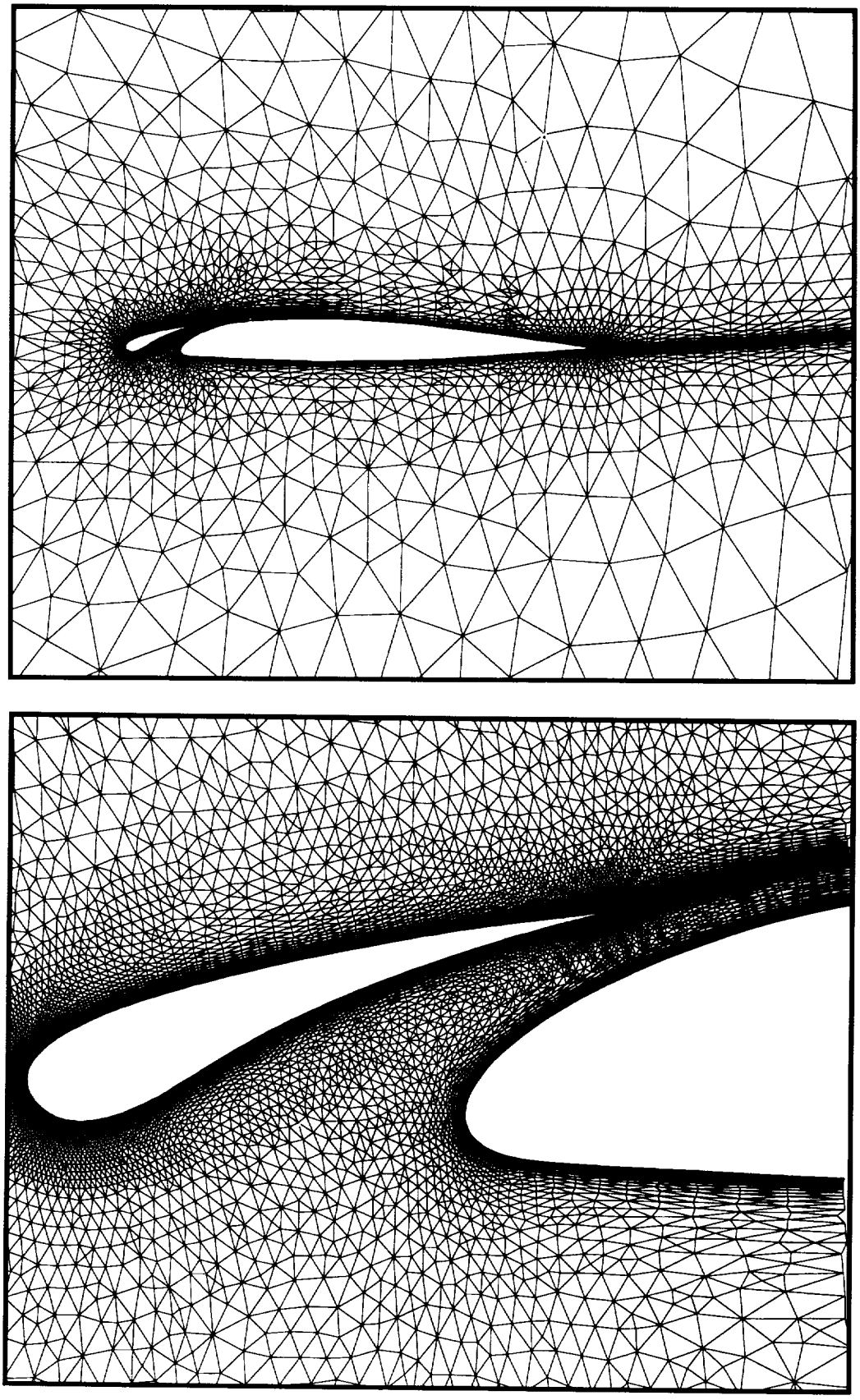

Figure 8
Global View of Coarse Unstructured Mesh and Close-Up View of Fine Unstructured Mesh Employed for Computing Flow Past a Two-Element Airfoil (Coarse Mesh Points = 7272, Fine Mesh Points = 28871)

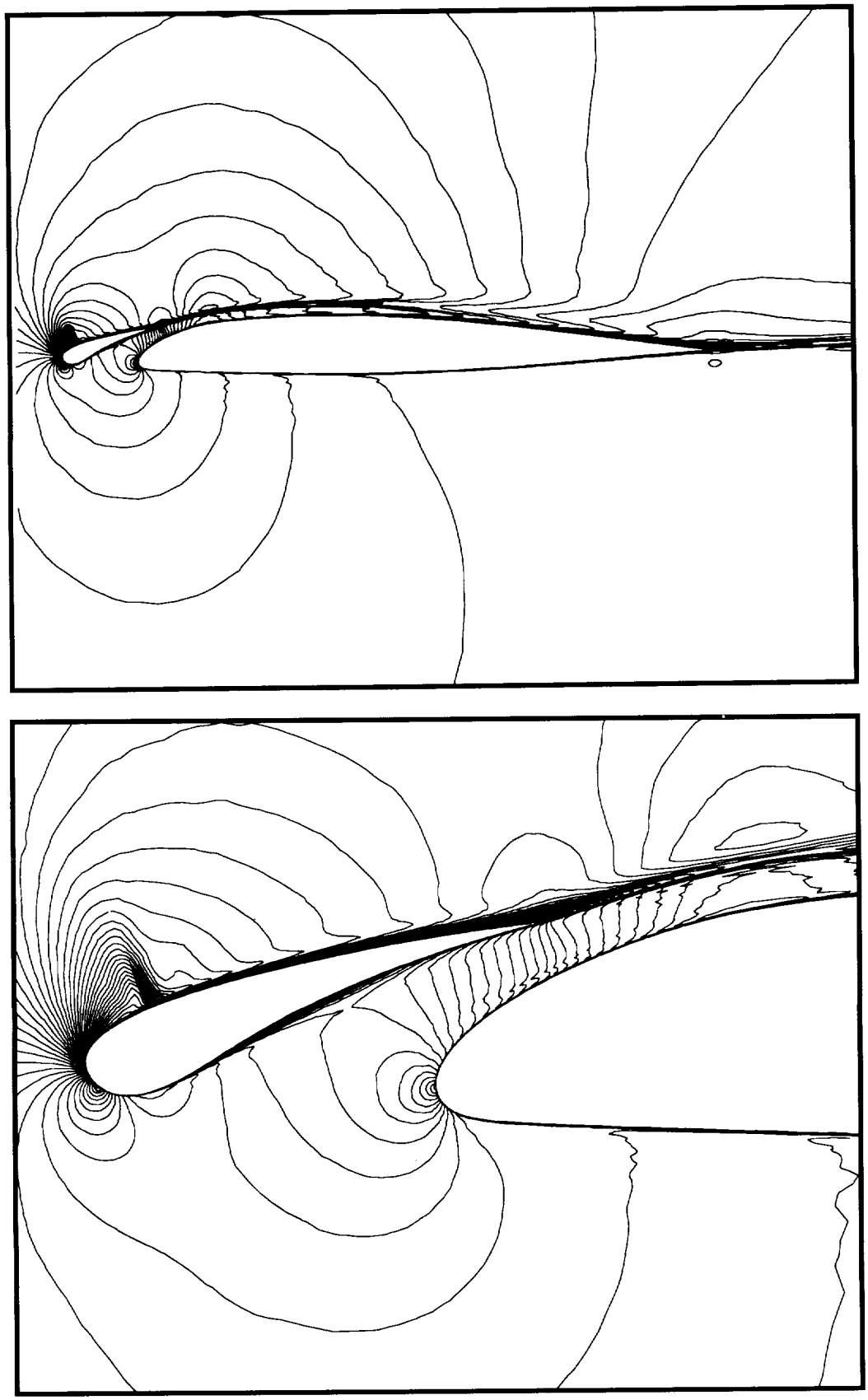

Figure 9
Computed Mach Contours Using Low-Reynolds Number Modification for Turbulence Equations for Supercritical Flow over a Two-Element Airfoil
(Mach = 0.5, Re = 4.5 million, Incidence = 7.5 degrees)

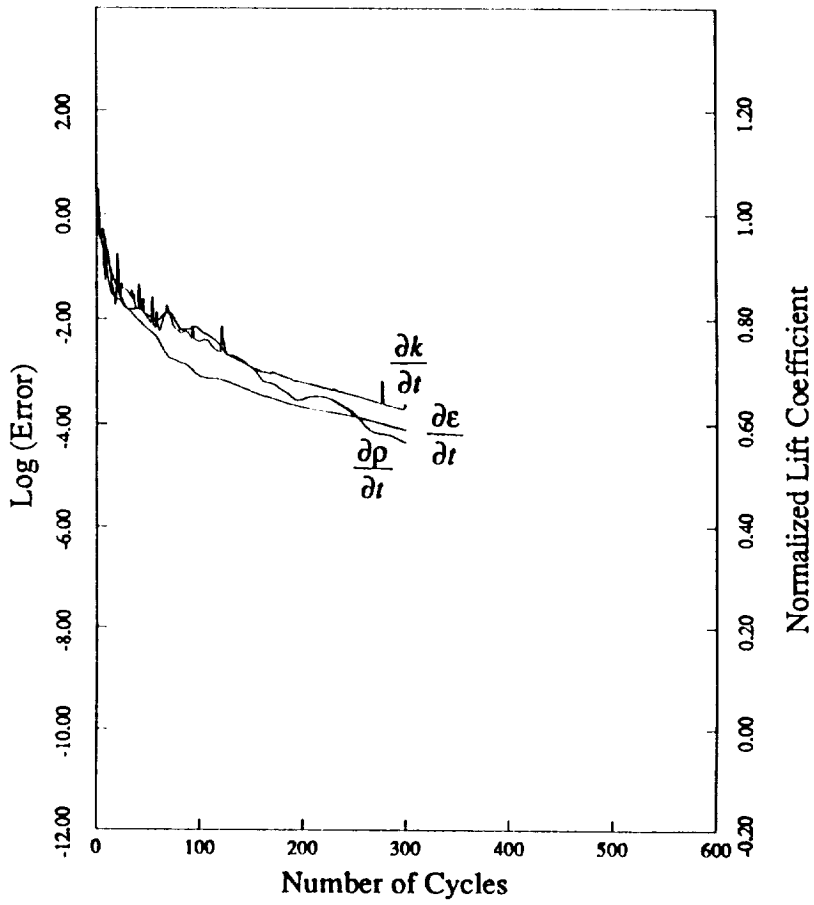

Figure 10
Multigrid Convergence Rate of the Density Equation and the Two Turbulence Equations Using Low-Reynolds Number Modifications for Flow Over Two-Element Airfoil (Mach = 0.5, Re = 4.5 million, Incidence = 7.5 degrees)

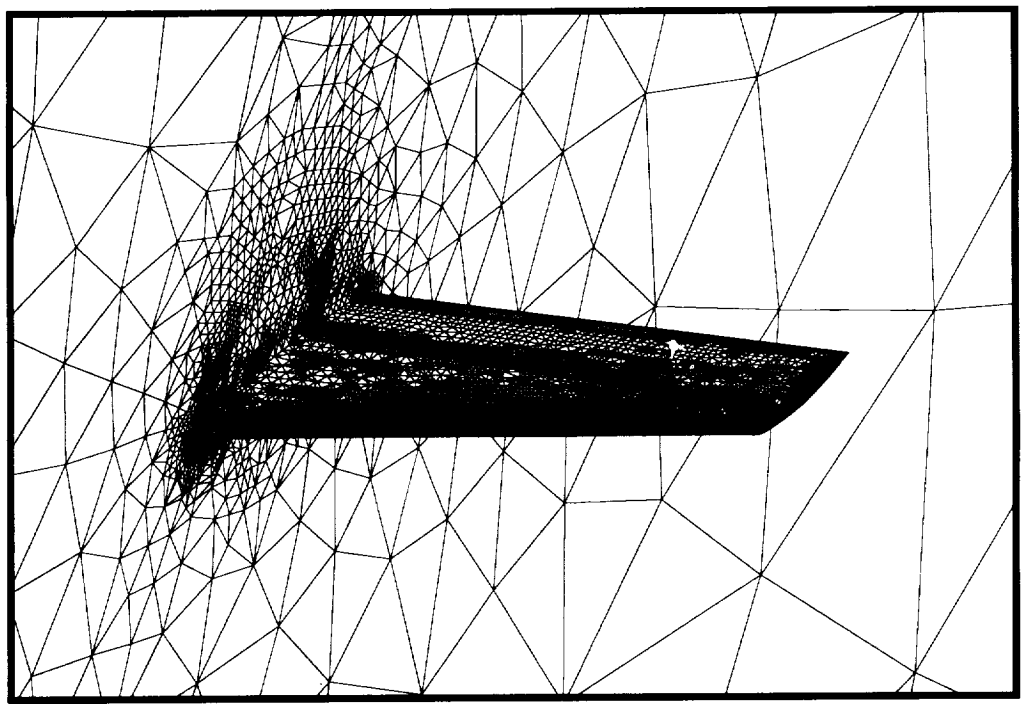

Figure 11
Finest Adapted Mesh Generated About ONERA M6 Wing
(Number of Nodes = 173,412 Number of Tetrahedra = 1,013,718)

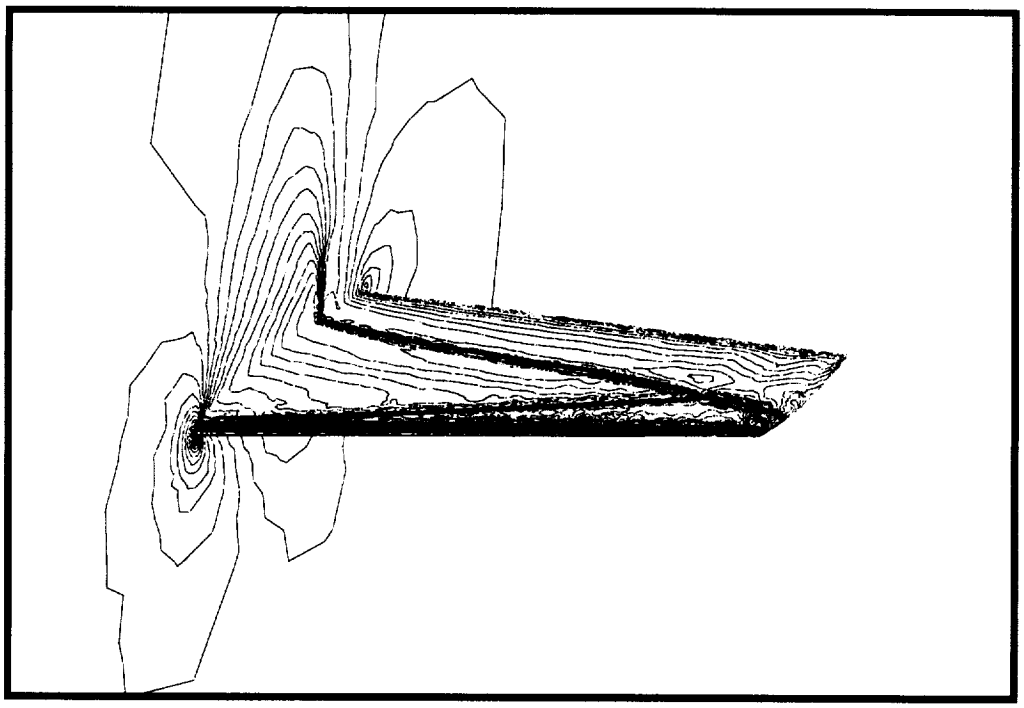

Figure 12
Computed Mach Contours on the Adaptively Generated Mesh About the ONERA M6 Wing
(Mach = 0.84, Incidence = 3.06 degrees)

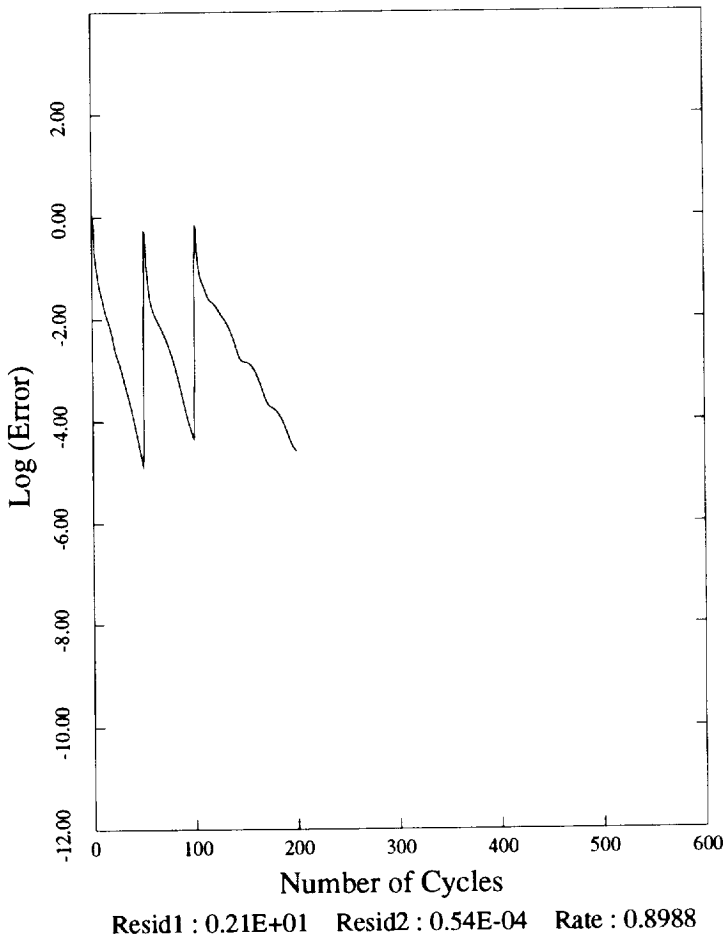

Figure 13
Convergence Rate of the Unstructured Multigrid Algorithm on the
Adaptively Generated Sequence of Meshes about the ONERA M6 Wing
as Measured by the Average Density Residuals Versus the Number of Multigrid Cycles

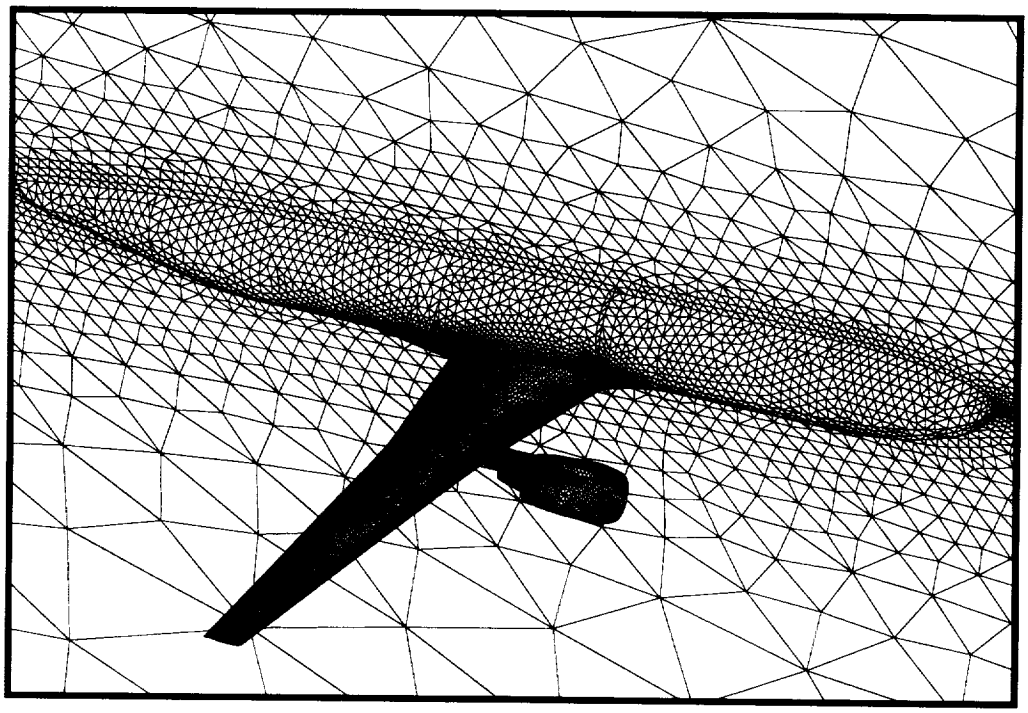

Figure 14
Coarse Unstructured Mesh about an Aircraft Configuration with Single Nacelle;
Number of Points = 106,064, Number of Tetrahedra = 575,986
(Finest Mesh Not Shown)

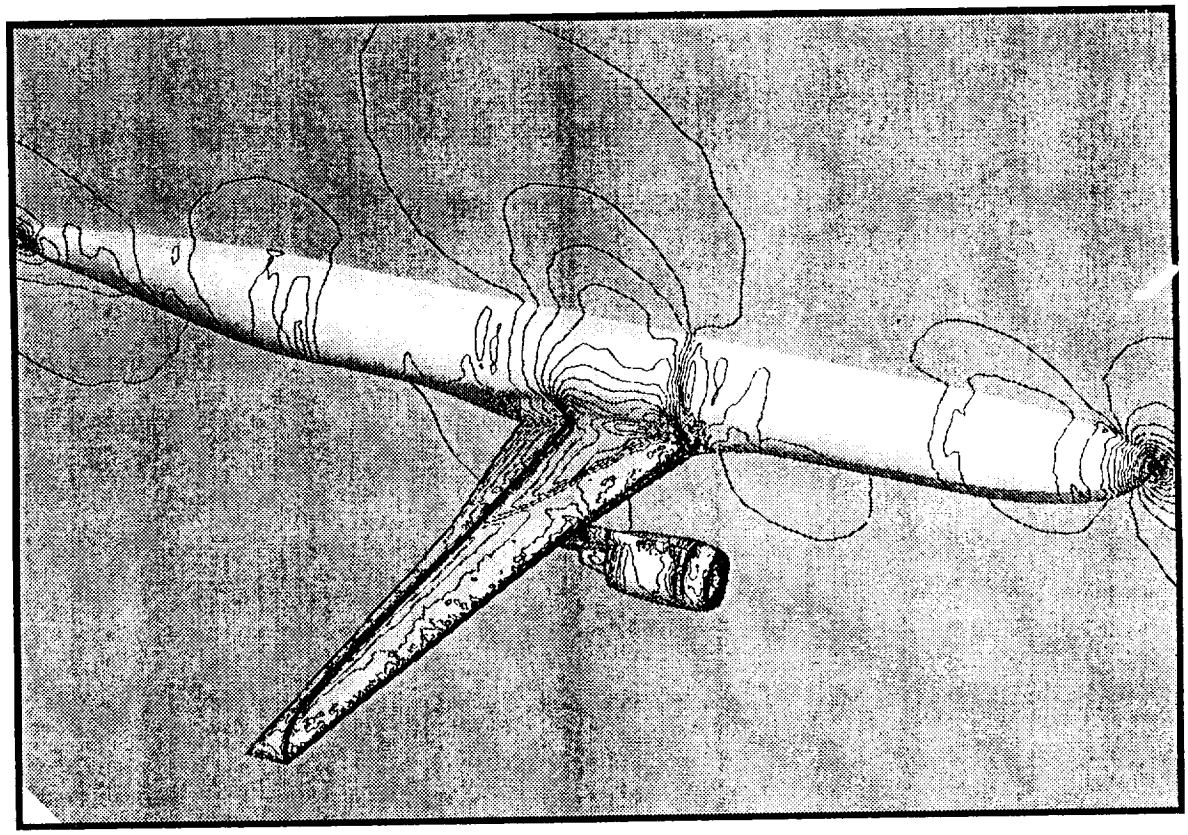

Figure 15
Mach Contours for Flow over Aircraft Configuration Computed
on Fine Mesh of 804,056 Vertices and 4.5 million Tetrahedra
(Mach = 0.768, Incidence = 1.116 degrees)

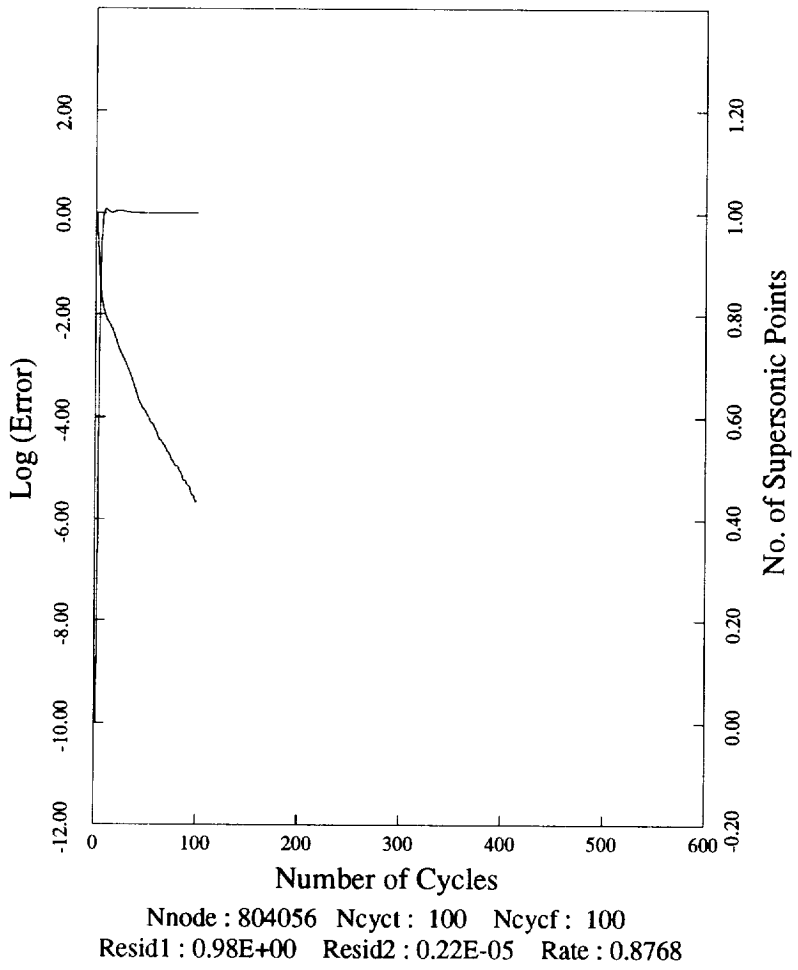

Figure 16
Multigrid Convergence Rate on Finest Mesh of the Multigrid Sequence for Transonic Flow over Aircraft-with-Nacelle Configuration

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1992	Contractor Report

4. TITLE AND SUBTITLE
UNSTRUCTURED MESH ALGORITHMS FOR AERODYNAMIC CALCULATIONS

5. FUNDING NUMBERS
C NAS1-18605
NAS1-19480

WU 505-90-52-01

6. AUTHOR(S)
D. J. Mavriplis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23665-5225

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 92-35

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-189685
ICASE Report No. 92-35

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Michael F. Card
Final Report

To appear in Proc. of the 13th Int. Conf. on Num. Methods in Fluid Dynamics (Springer-Verlag)

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 02, 64

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The use of unstructured mesh techniques for solving complex aerodynamic flows is discussed. The principle advantages of unstructured mesh strategies, as they relate to complex geometries, adaptive meshing capabilities, and parallel processing are emphasized. The various aspects required for the efficient and accurate solution of aerodynamic flows are addressed. These include mesh generation, mesh adaptivity, solution algorithms, convergence acceleration and turbulence modeling. Computations of viscous turbulent two-dimensional flows and inviscid three-dimensional flows about complex configurations are demonstrated. Remaining obstacles and directions for future research are also outlined.

14. SUBJECT TERMS
unstructured; grids; computational; aerodynamics

15. NUMBER OF PAGES
34

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std. Z39-18
298-102

NASA-Langley, 1992